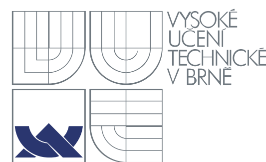










INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Vizualizace dynamických systémů v prostředí virtuální reality

Učební texty k semináři

Autor:

Ing. Jan Daněk (HUMUSOFT s.r.o.)

Datum:

20.1.2012

Centrum pro rozvoj výzkumu pokročilých řídicích a senzorických technologií CZ.1.07/2.3.00/09.0031

TENTO STUDIJNÍ MATERIÁL JE SPOLUFINANCOVÁN EVROPSKÝM SOCIÁLNÍM FONDEM A STÁTNÍM ROZPOČTEM ČESKÉ REPUBLIKY

OBSAH

1. ÚVOD

Cílem semináře je uvést účastníky do problematiky pokročilé vizualizace dynamických systémů. Vizualizace dynamických systémů je multi-disciplinárním oborem na pomezí řídicí techniky a počítačové grafiky. Pro odborníky v oblasti teorie řízení a inženýry existuje jistá bariéra při využití možností současných vizualizačních nástrojů. (Podobně existuje bariéra na straně grafiků a umělecké obce, kdy vizualizace realistického chování mnoha snadno popsatelných dynamických systémů je nejčastěji prováděna pracnými a primitivními interpolacemi pohybu jejich součástí.) Kromě vizualizace samotné přináší obousměrné spojení dynamických modelů soustav s prostředím virtuální reality celou řadu dalších zajímavých možností – možnost tvorby netradičních uživatelských rozhraní, modelování virtuálních haptických a telematických systémů, apod. Účastníci semináře se přehlednou formou seznámí s těmito možnostmi, na několika praktických příkladech si budou moci vyzkoušet vizualizaci systémů v prostředí MATLAB / Simulink / Simulink 3D Animation.

2. Jazyk VRML 97

2.1. Historie

Od té toby, co lidé začali publikovat informace v síti WWW, existovala snaha obohatit obsah WWW stránek o pokročilou interaktivní 3D grafiku.

Termín VRML – Virtual Reality Markup Language byl poprvé použit panem Timem Berners-Lee na Evropské konferenci o WWW v roce 1994, když mluvil o potřebě normy pro 3D obsah na Webu. Záhy se kolem diskusní skupiny www-vrml vytvořila aktivní skupina inženýrů a umělců. Tato skupina jazyk s cílem zdůraznit roli grafiky přejmenovala na Virtual Reality Modeling Language a v krátké době připravila specifikaci jazyka VRML 1. Jako základ pro první návrh byla použita podmnožina objektů formátu Inventor společnosti Silicon Graphics.

Norma VRML 1 byla implementována v několika VRML prohlížečích. Protože však tento jazyk podporoval jen tvorbu statických virtuálních scén, byly možnosti jeho rozšíření omezené. Brzy bylo jasné, že jazyk potřebuje rozšíření o možnost tvorby dynamických a interaktivních scén. Byla vyvinuta norma VRML 2, která byla v roce 1997 přijata jako mezinárodní norma ISO/IEC 14772-1:1997, od té doby se tomuto jazyku říká VRML97.

VRML97 je otevřená a flexibilní platforma pro tvorbu interaktivních 3D scén – virtuálních světů. S rapidním růstem výpočetního a grafického výkonu počítačů a rozvojem vysokorychlostních komunikačních sítí se rychle rozvíjelo využití 3D grafiky i v jiných než tradičních oblastech umění a her. V současnosti existuje celá řada prohlížečů VRML97 na různých platformách, integrovaných do WWW prohlížečů i samostatných aplikací. Rovněž existuje mnoho 3D editorů, CAD programů a dalších specializovaných grafických nástrojů, které jsou buď založeny na VRML97, nebo umožňují import a export modelů VRML97. V neposlední řadě je v tomto formátu na síti WWW k dispozici nepřeberné množství modelů různých objektů, které je možné (samozřejmě při respektování autorských práv) použít pro tvorbu vlastních scén.

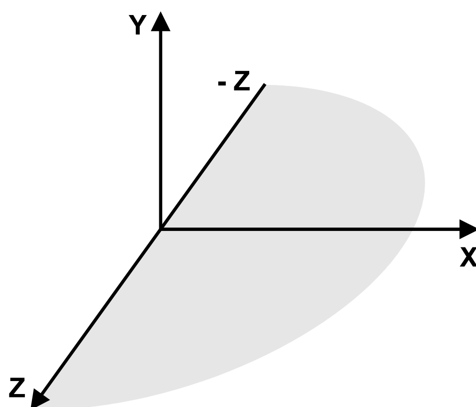
Organizace Web3D Consortium, která v současnosti zastřešuje iniciativy kolem otevřených formátů pro publikaci 3D obsahu v síti WWW, vydala v roce 2004

další normu, označovanou za nástupce jazyka VRML97 – X3D. Tento formát rovněž vychází z VRML, rozšiřuje jej o některé prvky a kromě klasického kódování nabízí i XML variantu. Formát X3D ovšem dosud není podporován zdaleka v takové míře jako VRML97, proto jsou možnosti jeho praktického využití prozatím omezené.

Vlastnosti jazyka VRML97, zejména jeho otevřenost, interaktivita, hierarchická struktura objektů a relativní jednoduchost jej předurčují k tomu, aby byl vynikajícím prostředkem pro pokročilou 3D vizualizaci v technických a vědeckých aplikacích. Proto byl také zvolen jako základní formát produktu Simulink 3D Animation™, nástroje pro vizualizaci dynamických systémů modelovaných v nástrojích MATLAB® / Simulink®.

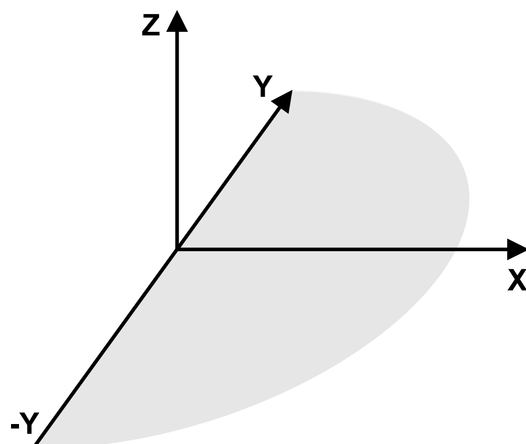
2.2. Souřadný systém VRML

VRML používá pravotočivý Kartézský souřadný systém. Jestliže palec, ukazovák a prostředník pravé ruky natočíme tak, aby mezi sebou svíraly pravé úhly, palec představuje osu +X, ukazovák osu +Y a prostředník osu +Z. Osa +X směřuje doprava, osa +Y nahoru a osa +Z směřuje k pozorovateli.



Obrázek 2.1 Souřadný systém VRML

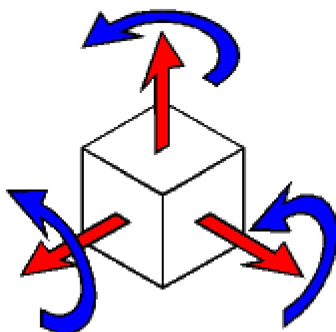
Souřadný systém VRML se liší od souřadného systému programu MATLAB, který je rovněž pravotočivý, ale osa +Y směřuje od pozorovatele a osa +Z míří nahoru. Rozdíly mezi těmito souřadnými systémy musíme brát do úvahy a při jakékoliv prostorové vizualizaci je potřebné souřadnice objektů převádět z jednoho systému do druhého. Jedná se o triviální transformaci, kde se prohodí pořadí 2. a 3. komponenty a u jedné z nich se změní znaménko.



Obrázek 2.2 Souřadný systém programu MATLAB

V mnoha případech se pro modelování fyzikálních soustav používá i další nástroj z rodiny programů MATLABu – SimMechanics. Tento nástroj využívá souřadný systém, který je totožný se systémem VRML, tělesa modelovaná s použitím SimMechanics lze tedy vizualizovat přímo bez jakékoliv transformace.

Rotace jsou ve VRML definovány s použitím notace "osa-úhel" a pravidla pravé ruky. Představte si, že zavřenou pravou rukou držíte osu otáčení tak, že zdvižený palec směřuje k jejímu kladnému konci. Pokud palec směřuje v kladném směru osy, zbývající čtyři prsty směřují proti směru hodinových ručiček. V tomto směru je orientován kladný úhel otočení objektu kolem této osy.



Obrázek 2.3 VRML rotace

Použité jednotky: Ve VRML jsou všechny velikosti a vzdálenosti definovány v metrech, úhly v radiánech a čas v sekundách.

V hierarchické struktuře objektů VRML platí, že pozice a rotace objektů, které jsou potomky určitého objektu, jsou definovány v lokálním souřadném systému tohoto nadřízeného objektu.

2.3. Krátký popis formátu souboru VRML

Přestože se virtuální světy vytvářejí nejčastěji s použitím grafických editorů, pro pochopení toho, jaké vlastnosti objektů je možné řídit a animovat je dobrá alespoň orientační znalost struktury VRML souborů. To vám také umožní efektivnější vytváření 3D scén.

Informace v této sekci jsou velmi zkráceny. Úplný popis struktury jazyka VRML97 naleznete v Normě VRML97 [1].

V jazyce VRML je 3D scéna popsána jako hierarchická stromová struktura objektů – uzlů. Každý uzel představuje určitý soubor vlastností objektů scény. Existuje 54 různých typů uzlů (Nodes). Některé představují skutečné 3D objekty – tvary, některé jsou určeny k jiným účelům, například k seskupení jim podřízených uzlů do jednoho nadřazeného objektu. Několik příkladů:

- Uzel **Box** představuje těleso tvaru kvádru
- Uzel **Transform** definuje pozici, měřítko, rotaci a další vlastnosti skupiny svých podřízených uzlů (seskupovací uzel)
- Uzel **Material** reprezentuje materiál povrchu objektů
- Uzel **DirectionalLight** je jedním z druhů světel, které je možné definovat
- Uzel **Fog** umožňuje definovat optické vlastnosti prostředí
- Uzel **ProximitySensor** je jedním z uzlů, pomocí kterých lze dosáhnout interaktivnosti virtuálních scén. Tento uzel generuje události, když uživatel (návštěvník virtuálního světa) vstoupí, vystoupí a pohybuje se uvnitř určitého definovaného prostoru

Každý uzel obsahuje seznam polí (Fields) obsahujících hodnoty, které definují parametry jeho funkčnosti.

Uzly mohou být umístěny na hlavní úroveň anebo jako potomci určitého uzlu v hierarchii objektů virtuálního světa. Pokud změníme určitou vlastnost některého uzlu, budou tím ovlivněni všichni jeho potomci. To nám umožňuje v některých případech snadno definovat vzájemné vztahy uvnitř složitých sestav.

Každý uzel může být pojmenován unikátním jménem. Syntaxe jazyka VRML má pro to vyhrazeno klíčové slovo **DEF**. Například příkaz:

DEF MyNodeName Box

definuje jméno tohoto objektu typu Box jako MyNodeName.

SIMULINK 3D ANIMATION UMOŽŇUJE ČÍST A NASTAVOVAT VLASTNOSTI POUZE TAKTO POJMENOVANÝCH UZLŮ!

V následujícím příkladě jsou v jednoduchém virtuálním světě pouze dva grafické objekty – podlaha reprezentována plochým kvádrem a nad ní červená kulička:

```
#VRML V2.0 utf8
# Toto je komentar
WorldInfo {
  title "Bouncing Ball"
}
Viewpoint {
  position 0 5 30
  description   "Side View"
}
DEF Floor Box {
  size 6 0.2 6
}
DEF Ball Transform {
  translation    0 10 0
  children Shape {
    appearance Appearance {
      material Material {
        diffuseColor 1 0 0
      }
    }
    geometry Sphere {
    }
  }
}
```

První řádek je řádek záhlaví VRML, každý soubor ve formátu VRML97 musí začínat tímto záhlavím. Záhlaví definuje, že se jedná o soubor VRML97 kódovaný podle normy UTF8. Znak “#” je vyhrazen k uvození komentářů – vše,

co je na řádku za tímto znakem, je prohlížeči ignorováno (s výjimkou prvního řádku záhlaví).

Většina vlastností uzlu **Box** je ponechána na jejich implicitních hodnotách – poloha v počátku souřadného systému, materiál, barva atd. Tomuto objektu jsou přiřazeny pouze jméno Floor a rozměry. U kuličky je definováno mnohem více vlastností. Aby bylo možno kuličkou pohybovat, je uzel **Shape**, který nese informaci o tvaru (podřízený uzel **Sphere** v poli geometry) umístěn jako podřízený uzel uzlu **Transform**. Červená koule o implicitním poloměru 1 m je umístěna 10 m nad podlahu. Dále je v tomto světě definován jeho název (obvykle je zobrazen v okně prohlížeče) a vhodný počáteční pohled.

Pokud budeme chtít vizualizovat polohu skákající kuličky ze Simulinku, budeme nastavovat hodnotu pole **translation** uzlu **Ball**.

2.4. Datové typy VRML

Vlastnosti uzlů jsou definovány pomocí jejich polí (fields), které mohou být různých datových typů. Tyto typy musíme znát, abychom z externí aplikace (například pomocí nástroje Simulink 3D Animation) mohli do virtuální scény posílat hodnoty správného typu.

Typy, které vyjadřují vlastnost jednoho prvku, jsou uvozeny prefixem **SF**, pole, v nichž je uložen seznam více hodnot stejného typu, jsou uvozena prefixem **MF**.

Jednoduchý typ	*Seznam*	*Význam*	*Příklad hodnot*
SFBool	---	logická hodnota	TRUE, FALSE
SFInt32	MFInt32	celé číslo v rozsahu 32 bitů	-42, 0, 123456
SFFloat	MFFloat	číslo s desetinnou tečkou	-3.14, .001, 55
SFTime	MFTime	čas v sekundách	0.0, 60
SFVec2f	MFVec2f	vektor v rovině	3 –2, 1 1, -.5 1.7
SFVec3f	MFVec3f	vektor v prostoru	3 –2 5, 1 1 0, -.5 1.7 –3.33
SFColor	MFColor	barva ve složkách RGB, každá z intervalu <0, 1>	0 0 0, 1 1 1, .7 .5 0.3
SFRotation	MFRotation	osa (vektor v prostoru) a úhel rotace (v radiánech)	0 0 1 1.57, -1 0 0 -3.14
SFNode	MFNode	uzel VRML	
SFString	MFString	textový řetězec	“Dobrý den.“
SFImage	---	strukturovaný vzor pixelů	

Konkrétní datový typ se u každého pole dozvíme v části 6. Node Reference normy VRML97.

2.5. Datové třídy VRML

Kromě datových typů jsou pole uzlů rozdělena ještě do tříd, které definují chování uzlů, způsob uložení uzlů v paměti počítače a jak uzly interagují s jinými uzly a s externími objekty. Uzel může obsahovat čtyři datové třídy: field, exposedField, eventIn and eventOut.

Datová třída	*Popis*
eventIn	Událost, kterou může uzel přijmout
eventOut	Událost, kterou může uzel vyslat
field	Soukromý člen uzlu
exposedField	Veřejný člen uzlu – kombinace eventIn a eventOut

Konkrétní datovou třídu se u každého pole dozvíme v části 6. Node Reference Normy VRML97.

3. Tvorba virtuálních světů

Formát VRML je založen na standardním textu, který můžeme číst a upravovat jakýmkoliv textovým editorem. Pokud použijeme správnou syntaxi jazyka VRML, můžeme virtuální scény vytvářet textovým editorem, podobně jako je tomu u stránek HTML. Mnoho lidí dává při tvorbě zejména jednoduchých scén přednost svému oblíbenému textovému editoru. Nicméně standardní způsob tvorby virtuálních scén je využití některého 3D grafického editoru. Tyto nástroje umožňují tvorbu i složitých virtuálních scén bez hluboké znalosti jazyka VRML. Grafické nástroje poskytují uživateli komfortní prostředí pro efektivní tvorbu nejrůznějších scén. Je rovněž běžné, že při tvorbě svých scén kombinujeme 3D objekty získané z různých zdrojů – například můžeme využít knihovny objektů daného editoru a importovat objekty z CAD nástrojů či od poskytovatelů 3D obsahu na síti WWW.

3.1. Nástroje pro tvorbu virtuálních scén

Pro tvorbu VRML virtuálních světů jsou k dispozici tyto nástroje a služby:

- Nativní VRML editory
- Všeobecné 3D editory, které podporují export do formátu VRML97
- Specializované 3D nástroje
- Nástroje pro konverzi a optimalizaci modelů
- Poskytovatelé 3D obsahu

3.1.1. Nativní VRML editory

Nativní VRML editory používají VRML jako svůj hlavní, nativní formát. To je zárukou, že všechny vlastnosti poskytované editorem jsou kompatibilní s jazykem VRML. Nativní VRML editory také mohou využít všechny unikátní vlastnosti jazyka VRML, jako jsou interpolátory, senzory, využití konstruktu USE apod. Bohužel jen několik málo pokročilých editorů dosáhlo komerčního rozšíření a kvality. Jeden z nich, V-Realm Builder, je součástí distribuce produktu Simulink 3D Animation. Další nativní VRML editory jsou:

- Flux Studio

- Internet Scene Assembler
- SwirlX3D
- VrmlPad

3.1.2. *Všeobecné 3D editory*

Všeobecné 3D editory nepoužívají VRML jako svůj základní formát, ale nabízejí možnost exportu vytvořených modelů do formátu VRML. Existuje mnoho grafických nástrojů s podporou VRML, například tyto:

- 3ds Max
- AC3D
- Autodesk Maya
- Blender
- CATIA
- Google SketchUp
- LightWave
- modo
- Pro/ENGINEER
- SolidWorks

Všeobecné 3D editory jsou zaměřeny na určitý typ práce. Například vizuální umění, hry, CAD aplikace. Nabízejí různá pracovní prostředí přizpůsobená svému účelu. Některé 3D nástroje jsou velmi mocné, nákladné a složité, jiné jsou poměrně levné nebo dokonce poskytované zdarma, přesto mohou uspokojit potřeby některých uživatelů.

Při použití těchto nástrojů musíme obvykle vzít do úvahy rozdíl mezi jejich nativním formátem a VRML a konverzi do VRML provést s uvažováním těchto rozdílů. Například, při exportu parametrických modelů do formátu VRML je obvykle vhodné provést úvahu o správném nastavení přesnosti teselace (triangularizace) VRML modelu tak, aby byla při odpovídajícím výkonu renderování modelu co nejlépe zachována jeho přesnost.

3.1.3. Specializované 3D nástroje

3D nástroje, které vynikají v určité specifické oblasti tvorby 3D objektů. Patří sem nástroje pro modelování zemského povrchu, městských aglomerací, skenování pomocí laseru, reverzní 3D engineering apod.

- 3DEM
- World Construction Set / Scene Express
- FARO Laser Scanner software

3.1.4. Nástroje pro konverzi a optimalizaci modelů

Mnoho všeobecných 3D editorů může exportovat své modely do VRML. Také nativní VRML editory mohou často importovat soubory vytvořené všeobecnými nástroji. Jsou však i nástroje, které se specializují na konverzi modelů. Tyto nástroje mají obvykle nejlepší podporu různých formátů. Navíc často poskytují optimalizaci a selektivní redukci počtu polygonů ve scéně. Jsou to například tyto nástroje:

- PolyTrans
- Deep Exploration
- VizUp

3.1.5. Poskytovatelé 3D obsahu

Efektivní způsob, jak získat některé komponenty virtuálních scén, je zakoupit je u organizace, která se specializuje na poskytování 3D modelů objektů. S použitím VRML editoru pak můžeme rychle poskládat velmi zajímavé virtuální scény. Na síti WWW existuje mnoho zdrojů 3D modelů. Některé populární zdroje jsou uvedeny níže:

- www.turbosquid.com
- www.3dexport.com
- www.the3dstudio.com
- www.3d02.com
- www.top3d.net/

- sketchup.google.com/3dwarehouse

4. PROPOJENÍ VIRTUÁLNÍCH SVĚTŮ S DYNAMICKÝMI MODELY

Jazyk VRML umožňuje vytvářet dynamické a interaktivní scény. S pomocí senzorů, interpolátorů, skriptů a časovačů je možné vytvořit celou řadu působivých efektů. Nicméně existuje mnoho případů, kdy je vhodné virtuální scény propojit s dynamickými modely realizovanými v jiných aplikacích. Ze strany tvůrců virtuálních světů se jedná například o zahrnutí věrného pohybu respektujícího skutečnou dynamiku těles. Mnohem širší využití má však toto propojení ze strany odborníků zabývajících se simulací dynamických systémů. Vizualizace v prostředí virtuální reality může přispět k lepšímu porozumění, jak systémy fungují, je možné vytvářet 3D uživatelská rozhraní k ovládání systému, a podobně. Zatím málo využívanou oblastí je uzavření zpětnovazební smyčky řídicích systémů přes virtuální scénu, či použití virtuálních scén jako zdroje 2D videosignálu u systémů pro zpracování obrazu.

Možnosti propojení virtuálních scén s dynamickými modely si ukážeme s využitím produktu Simulink 3D Animation v prostředí MATLAB a Simulink.

4.1. Vizualizace modelů v Simulinku pomocí Simulink 3D Animation

Na příkladě si ukážeme, jak model v Simulinku napojit na virtuální svět. Předpokládejme, že jsme si připravili simulační model vzletu letadla, který chceme vizualizovat v prostředí virtuální reality.

Jednoduchý model vzletu letadla je připraven jako standardní demonstrační příklad **vrtut2**. Spusťte Matlab a v příkazovém řádku zadejte **vrtut2**. Otevře se model (prozatím bez bloků, které zajišťují 3D vizualizaci).

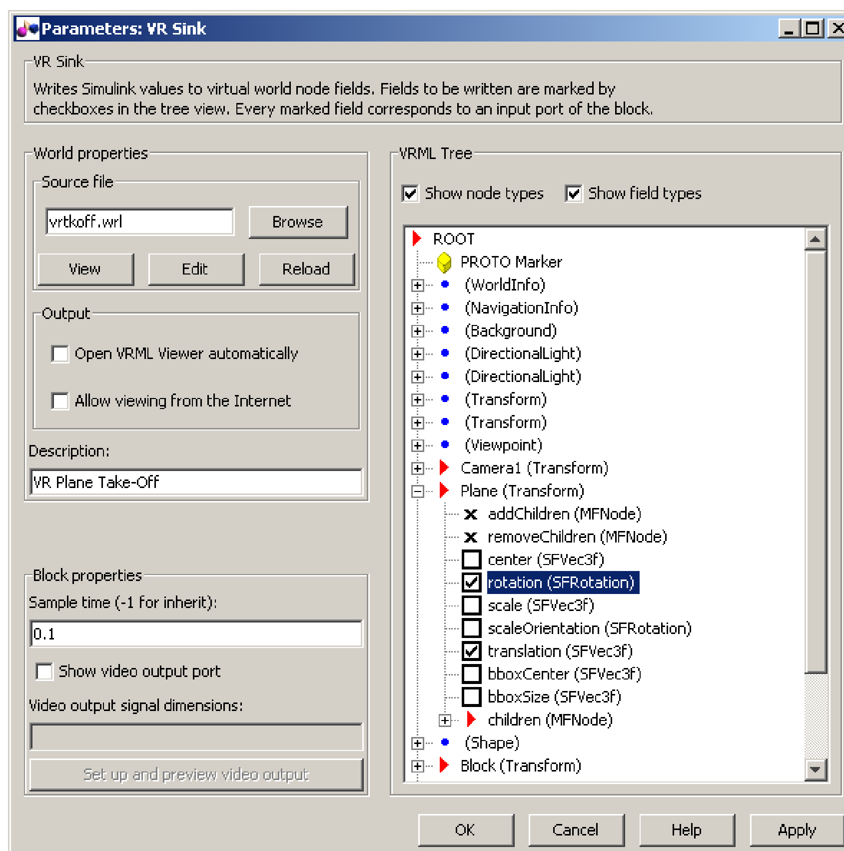
Můžete spustit simulaci modelu a pozorovat průběh zajímavých veličin pomocí osciloskopů.

Otevřete knihovnu bloků Simulink 3D Animation – zadejte příkaz **vrlib**. Do svého modelu přetáhněte blok **VR Sink**.

Nyní je potřeba vytvořit asociaci mezi blokem VR Sink a virtuální scénou. Jednoduchá virtuální scéna s letadlem a vzletovou a přistávací dráhou je připravena v souboru vrtkoff.wrl.

Otevřete blok VR Sink. A window similar to that on Figure 10 appears.

V sekci **Source file** klepněte na tlačítko Browse a vyberte soubor vrtkoff.wrl. Všimněte si, že se vyplní pole Description (na základě informace v poli Title uzlu WorldInfo ve VRML souboru). V pravé části dialogu se objeví strom představující zjednodušenou strukturu právě asociovaného virtuálního světa. U uzlů, jejichž vlastnosti je možné ovládat, je červený trojúhelník. Tyto uzly je možné rozkliknout, zpřístupní se nám buď seznam polí, jejichž hodnoty můžeme nastavovat ze Simulinku, anebo seznam podřízených uzlů. Po rozkliknutí uzlui Plane se nám zobrazí seznam vlastností, které můžeme u tohoto objektu (typu Transform) nastavovat. V tomto případě nás budou zajímat pole **translation** a **rotation**, představující polohu a natočení letadla. Zaškrtněte políčka u těchto polí. Dialog nybní vypadá podobně jako na následujícím obrázku.



Obrázek 4.1 Dialog bloku VR Sink

Klepněte na OK. Blok VR Sink má teď dva vstupy, které musíme napojit na příslušné signály Simulinku.

Druhý vstup odpovídá poli **translation** objektu Plane. Toto pole představuje polohu letadla ve virtuálním světě. Dialog bloku VR Sink nám napovídá, že toto pole je typu SFVec3f. Poloha je definována třísložkovým vektorem [X Y Z]. V našem případě je virtuální scéna připravena tak, že letadlo vzlétá na vzletové dráze podél osy Z, výška letadla určuje jeho Y souřadnici. Dynamický model vzletu rovněž uvažuje jen tyto dvě souřadnice – třetí souřadnici nemáme k dispozici (v rámci modelu se nemění, model sleduje jen pohyb v rovině YZ). Přesto musíme do virtuálního světa poslat vektor o třech složkách. To zajistí blok **VR Signal Expander**, který vstupní vektor doplní na potřebný rozměr signály s hodnotou **VR Placeholder.** Tato hodnota je výpočetním strojem Simulink 3D Animation interpretována jako „tuto souřadnici vstupního vektoru neměň, použij hodnotu definovánu v souboru VRML“. Výstup bloku VR Signal Expander napojte na vstup Plane.translation. Všimněte si, že v dialogu bloku VR Signal Expander je prohozené pořadí 2. a 3. souřadnice (v poli Output signal indices je definováno [3 2]). Tak je zajištěna konverze souřadnic MATLABu na souřadnice VRML.

První vstup odpovídá poli **rotation** objektu Plane. Toto pole představuje natočení letadla. Dialog bloku VR Sink nám napovídá, že toto pole je typu SFRotation. Rotace je definována vektorem o 4 složkách. První tři složky definují osu rotace. V našem případě modelujeme jen jeden polohový úhel letadla – podélný sklon (elevaci), což je při jeho vzletu podél osy Z natočení kolem osy X. Jako osu rotace tedy definujeme vektor [1 0 0]. Poslední složkou tohoto vektoru je úhel sklonu v radiánech. Vstup **Plane.rotation** napojte na výstup příslušného multiplexeru.

Po napojení signálů ještě jednou otevřete blok VR Sink. Všimněte si, že jakmile je s tímto blokem asociován určitý virtuální svět, otevření tohoto bloku nezobrazí jeho dialog s parametry, ale otevře příslušnou virtuální scénu ve VR prohlížeči. K parametrům bloku se dostaneme po klepnutí na tlačítko se standardní ikonou **Block Parameters** v okně prohlížeče.

4.2. Simulink 3D Animation – využití rozhraní MATLABu

Základním způsobem práce s nástrojem Simulink 3D Animation je využití jeho Simulinkové knihovny, jak je popsáno výše. Simulink 3D Animation však umožňuje interakci s virtuálními světy i prostřednictvím rozhraní MATLABu. Virtuální světy můžeme řídit z příkazového řádku programu MATLAB nebo s použitím M-souborů.

Abychom mohli pracovat s virtuálním světem, musíme nejdříve vytvořit a otevřít objekt typu **vrworld**. Otevřený svět můžeme zobrazit, procházet jeho strukturu, číst a nastavovat parametry uzlů v něm obsažených. Ke čtení a nastavování hodnot polí u uzlů virtuálního světa slouží metody objektu **vrnode**. Na konci práce s virtuálním světem je vhodné objekt vrworld uzavřít a smazat.

Úplný seznam funkcí a metod rozhraní MATLABu naleznete v dokumentaci produktu Simulink 3D Animation, zde uvedeme jen krátký příklad.

Vytvořme objekt virtuálního světa asociovaného s jednoduchým VRML souborem 'vrmount.wrl':

```
wh = vrworld('vrmount.wrl');
```

V proměnné **wh** je nyní uložen handle k vytvořenému objektu typu vrworld.

Nyní otevřeme tento objekt, čímž jej zpřístupníme další interakci z MATLABu:

```
open(wh);
```

Virtuální svět zobrazme v okně prohlížeče Simulink 3D Animation:

```
view(wh);
```

Nyní je možné zjistit strukturu virtuálního světa. Objekt **vrworld** obsahuje několik objektů typu **vrnode**, které odpovídají uzlům ve VRML souboru pojmenovaným s pomocí VRML příkazu **DEF**.

Nejdříve si můžeme zobrazit seznam uzlů v daném virtuálním světě:

```
nodes(wh);
```

```
View1 (Viewpoint) [VR Car in the Mountains]
Camera_car (Transform) [VR Car in the Mountains]
VPfollow (Viewpoint) [VR Car in the Mountains]
Automobile (Transform) [VR Car in the Mountains]
```

```
Wheel (Shape) [VR Car in the Mountains]
Tree1 (Group) [VR Car in the Mountains]
Wood (Group) [VR Car in the Mountains]
Canal (Shape) [VR Car in the Mountains]
ElevApp (Appearance) [VR Car in the Mountains]
River (Shape) [VR Car in the Mountains]
Bridge (Shape) [VR Car in the Mountains]
Road (Shape) [VR Car in the Mountains]
Tunnel (Transform) [VR Car in the Mountains]
```

V našem virtuálním světe je celá řada pojmenovaných uzlů, my bychom rádi animovali polohu auta – uzlu **Automobile**. Tento uzel je typu **Transform** – jeho polohu tedy můžeme nastavit změnou hodnoty jeho pole **translation**. (Počáteční polohu auta zjistíme například tak, že stiskneme F5 – zobrazíme si drátěný model virtuálního světa – auto se skrývá v tunelu nalevo.)

K polím tohoto uzlu můžeme přistupovat dvojím způsobem – buď s použitím tečkové notace, anebo tak, že získáme handle k jeho objektu vrnode a použijeme metod tohoto objektu:

Tečková notace:

```
pos = wh.Automobile.translation

3.0000      0.2500      0
```

Vytvoření objektu vrnode:

```
nh = wh.Automobile;

pos = nh.translation
```

<nebo také>

```
pos = getfield(nh, ‘translation’)

3.0000      0.2500      0
```

Nyní nastavme novou polohu auta. Můžeme to udělat hned třemi způsoby:

```
nh.translation = [3 0.25 10];

wh.Automobile.translation = [3 0.25 10];

setfield(nh, ‘translation’, [3 0.25 10]);
```

Předtím, než zadáte jeden z těchto příkazů, si uspořádejte okna MATLABu a virtuálního prohlížeče tak, aby se nepřekrývala, abyste mohli pozorovat jeho efekt. Vidíme, že auto vyjelo z tunelu.

Na konci práce s objektem vrworld je správné tento objekt korektně uzavřít a smazat z paměti počítače:

```
close(wh);

delete(wh);
```

5. Using CAD Models with Simulink 3D Animation

When working with models of dynamic systems, it is an often requirement to visualize them in an advanced three-dimensional, preferably virtual reality environment. As most of the 3D designs in companies are created using some CAD tools, users need to convert these designs into a form that can be used in connection with Simulink® or SimMechanics® models and MATLAB®-based applications.

This section describes how to adapt existing CAD designs for visualization in the virtual reality environment using Simulink 3D Animation™. The process usually consists of three steps:

- **Exporting VRML models from CAD tools**
  - Converting part or assembly model into the VRML format used by the Simulink 3D Animation
- **Virtual scene modeling**
  - Composing the converted model into a virtual scene like urban or manufacturing environment, adding other objects important for interaction with the scene – viewpoints, background, lights, etc.
- **Linking the virtual scene to a Simulink / SimMechanics / MATLAB model**
  - Converting dynamic model quantities represented by Simulink signals, SimMechanics quantities or MATLAB data objects into form corresponding to VRML object properties (positions, rotations, etc.) and establishing a live data connection between the model and the virtual scene.

5.1. Exporting VRML models from CAD tools

The first part of the conversion process is exporting the CAD model of a part or whole product assembly from the CAD tool into the VRML format used by the Simulink 3D Animation. Most CAD tools have VRML export filters available. In addition, there are conversion utilities available from 3rd parties that can perform this task when the export filter isn't available in the CAD tool directly.

When exporting CAD models into VRML format, usually several options can be set to customize the output, either as options specific to export filters, or as general CAD document properties. Most usual and useful are the following:

- VRML format type
- Level of detail of exported files
- Units used in exported files
- Coordinate system used
- Assembly hierarchy

Consult your CAD system documentation for details on setting these properties.

5.1.1. VRML Format Type

There are two major versions of the VRML format used by graphic tools – older format **VRML1** and newer format called **VRML2**, or more often **VRML97**, according to the year of the ISO standard adoption. Simulink 3D Animation uses VRML97 standard format, so select the **VRML97** as the export format.

Hint: If your CAD tool allows only VRML1 format export, you can use the V-Realm Builder, a native VRML scene editor supplied with the Simulink 3D Animation, to convert models from VRML1 format to VRML97. Simply open a VRML1 file and resave it in V-Realm – the file is automatically saved in the VRML97 format.

All references to general abbreviation VRML here refer to the VRML97 standard.

5.1.2. Level of Detail Considerations

CAD models are usually parametric models that use proprietary methods for object rendering in various contexts. During VRML model export, the internal parametric model of the assembly is tessellated –model surface is divided into triangular meshes (represented in VRML by the **IndexedFaceSet** nodes). At this time, it is important to set the granularity of the mesh so that it is suitable for further use. Model simplification – polygon reduction later on would be very

difficult, as the full information about the object shape and structure is lost and cannot be reconstructed based on the (however complex) tessellated model.

For effective rendering of moving parts, it is recommended that VRML models are as simple as possible. On the other hand, no or little visible model degradation is usually required. We need to find a compromise between these two requirements.

As there are significant performance differences among various computers and graphic accelerators, there is no firm recommendation as to number of polygons / triangles suitable for use with Simulink 3D Animation. In order to assess the model complexity, it is advised to display the resulting VRML file in the Simulink 3D Animation viewer and observe the viewer response to navigation. If you can navigate in the virtual scene without any significant delay, the model is usually suitable for further work. Later on, when we connect the virtual scene to a Simulink model, there will be available a more exact measure of suitability – number of frames that are rendered per second during simulation.

5.1.3. Units Used in Exported Files

VRML length units are meters. In order to scale exported parts correctly in the virtual scene, it is suitable to export parts using meters. However, if the exported objects are very small or very large, which means that our planned virtual scene will be created to some scale, it is suitable to export objects using different units.

Note: VRML viewers are made to count with dimensions that are commensurate to us – people, in order to achieve the immersion effect of virtual reality graphics. Viewers assume that the scene author prepares the scene so that it can be walked through or examined by a virtual visitor to the scene (sometimes called the Avatar), whose physical dimensions are used in calculations for various purposes like collision detection, near object clipping or terrain following. Avatar dimensions (and also other navigation-specific parameters like default navigation speed) can be customized using the **NavigationInfo** VRML node. Simulink 3D Animation viewer is developed with the aim that effective navigation in the virtual scene is possible also in scaled

scenes (like inspecting miniature objects or visualizing a large-scale aircraft operation space), provided the scene author defined the NavigatioInfo parameters correctly.

5.1.4. Coordinate System Used

VRML uses right-handed Cartesian coordinate system, with axes defined so that:

+x points right

+y points up

+z points out of the screen

In order to avoid necessity to transform object axes into VRML system later on, it is good to export CAD models using identical coordinate system whenever possible. If your CAD tool uses a different coordinate system, and it doesn't allow you to change it for the exported objects, note the difference between the systems so that you can implement axes transformations in your model.

Make a note of the orientation of the parts in the coordinate system. For instance, a vehicle model exported so that it points towards the +x axis will have to be placed on a road in the virtual scene that starts in that direction. Model of vehicle dynamics will also need to count with this initial orientation of the vehicle 3D model.

When the CAD tool allows you to animate parts and assemblies, reset their positions to the initial state before the export.

5.1.5. Assembly Hierarchy

When exporting an assembly of several parts, there are usually two approaches possible:

- All parts are independent from each other

  Objects in the scene are independent from each other, at the same level of scene hierarchy. VRML file has a flat structure. All part coordinates are defined in global coordinates.

- Parts follow some kind of hierarchy defined in the CAD tool

  VRML file respects the hierarchy using the VRML Transform - children mechanism, has a nested structure. In this case, part coordinates are usually defined in part's parent local coordinate system.

  Example:

A robot can be exported with the following object hierarchy, in which each part coordinates are defined in the parent local coordinate system:

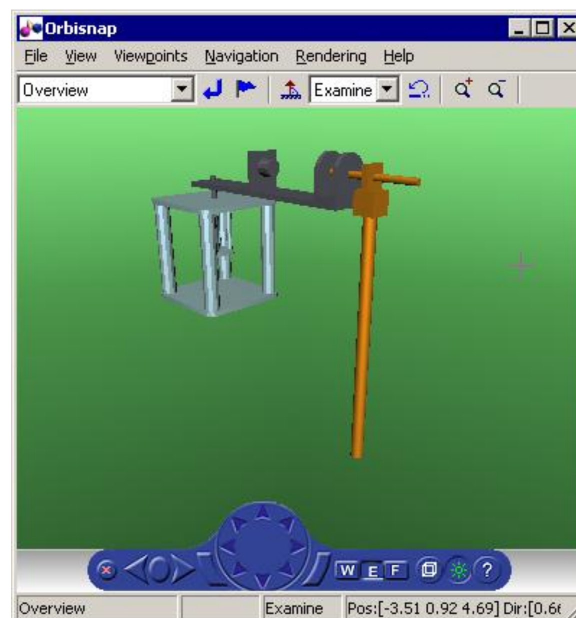
`rotating support - arm - wrist - hand – tool`

When **`rotating support`** moves, all other parts move with it, etc.

The hierarchy of the VRML file must correspond to the coordinates used in the dynamic model of the assembly:

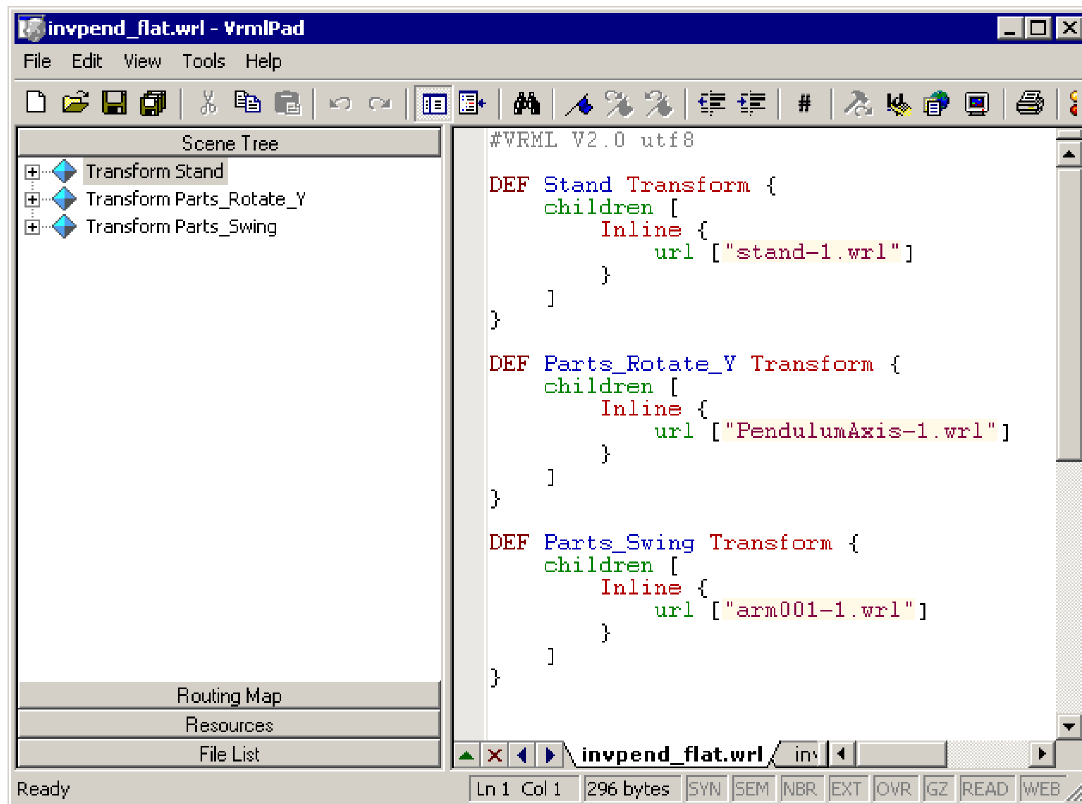
- If all parts in Simulink / SimMechanics are defined in global coordinates, we need a flat virtual scene structure
- If parts in Simulink / SimMechanics model follow hierarchical relation, we need a nested virtual scene structure

To illustrate these two approaches, let's imagine a rotation pendulum according to the following picture. The gray rotating arm rotates about vertical axis, the orange pendulum arm swings about the horizontal – z axis in rotating arm local coordinates:



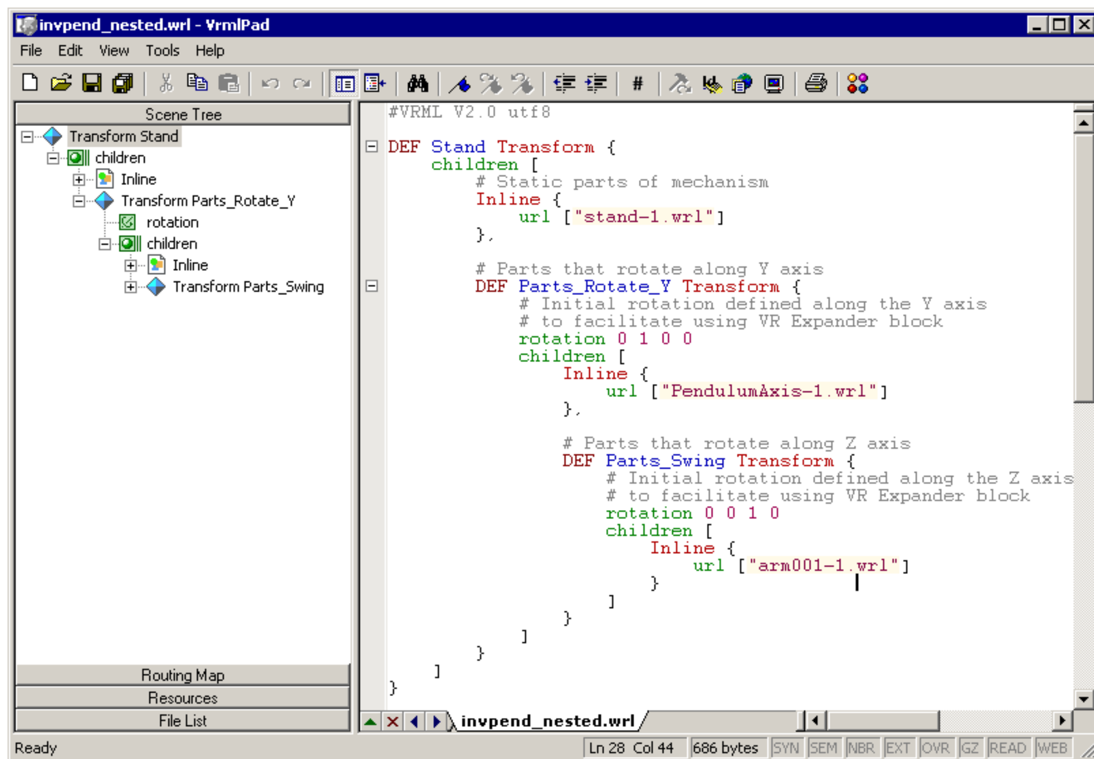
Obrázek 5.1 Virtual scene of rotating pendulum

When the pendulum dynamics model uses global coordinates for all moving parts, the VRML model has a flat structure as follows:



Obrázek 5.2 Flat structure of virtual scene

When the pendulum dynamics model uses local coordinates for moving parts, the corresponding VRML model needs to have a nested structure:



Obrázek 5.3 Hierarchical structure of virtual scene

Hint: Some tools allow you to export each part of the assembly into a separate VRML file. All parts are then referenced in a main file using the VRML **Inline** mechanism. This is a recommended way of working with assemblies – the main file is small in size, easy to understand and modify.

5.2. Virtual Scene Modeling

The results of CAD tool export filters often need few manual changes before being used by Simulink 3D Animation. Then we can create a scene in which we place the exported objects for further use.

5.2.1. Manual Modifications of Exported VRML Files

Manual adjustment of the exported VRML files can be done easily using a plain text editor. You can also use V-Realm Builder supplied with Simulink 3D Animation, but here we will explain the modifications made using a text editor. Usually, the following manual changes are necessary:

- Wrapping shape objects with Transforms
- Adding DEF names

5.2.2. Wrapping Shape Objects with Transforms

CAD tools export parts into VRML as individual shapes using various VRML object types. It is for instance possible to use a VRML **Shape** node or **Inline** mechanism. In order to be able to control part positions / orientations, we need to wrap each such **Shape** or **Inline** node by a node that allows changing these properties. It is a **Transform** node, whose purpose is to transform coordinates of its children. For instance, after wrapping by a Transform node, such **Inline** node will look like this:

```
Transform {
      children [
            Inline {
                  url ["robot_arm1.wrl"]
            }
      ]
}
```

In order to be able to place the whole assembly to the right initial position in the virtual scene, it is always good to wrap all parts of the assembly by an additional Transform node.

5.2.3. Adding DEF Names

CAD export filters often export objects with no names, or with synthetic, non-descriptive names. In order to be accessible from MATLAB, each VRML object needs to be given a unique name in the VRML file. This is achieved by adding a "**DEF Object_Name**" statement to the Transform line. After adding the DEF name, robot arm1 definition in the main VRML file looks like this:

```
DEF Robot_Arm1 Transform {
      children [
            Inline {
                  url ["robot_arm1.wrl"]
            }
      ]
}
```

These object names are used in Simulink 3D Animation functions and user interface. For instance, they appear as descriptions of inputs to **VR Sink** block. Therefore, it is good to give the parts descriptive names that help us with orientation in the object hierarchy.

Note: Sometimes it is also necessary to correct bugs introduced in the file by the CAD tool export filter. As VRML format is a text-based format codified by an ISO standard, these bugs are relatively easy to identify and correct.

5.2.4. Creating a Virtual Scene

The VRML file, adjusted manually in the previous steps, is ready for association with Simulink or SimMechanics models. However, in order to work with the virtual scene effectively, it is suitable to make some additional modifications to the scene file. These changes can be added on an ongoing basis, in parallel with developing and using the dynamic model.

These modifications can be done using a text editor, V-Realm or any other VRML editor:

- Adding the **WorldInfo** node with scene title (used as virtual world description in Simulink 3D Animation)
- Adding the **NavigationInfo** node defining the scene default navigation speed and avatar size that ensures correct displaying the object from near and far distances.
- Adding the **Background** node to specify a colour backdrop that simulates ground and sky, as well as optional background texture – panorama for the scene.
- Adding several viewpoints to be able to observe the object conveniently from different positions. The viewpoints can be static (defined as independent objects at the top level of the scene hierarchy) or attached to objects that will move in the scene to be able to observe them during simulation. Such viewpoints are defined as siblings of moving objects in the scene hierarchy. An example of a viewpoint moving with the object can be the viewpoint **`Ride on the Plane`** in the Simulink 3D Animation demo file **vrtkoff.wrl**.
- Adding lights to the scene in order to illuminate it. Although VRML viewers have always a “Headlight” available, it is a good practise to define lights in the scene so that it looks for every user in a consistent way, according to the scene author preferences. Most useful type of VRML light to illuminate a whole scene is the **DirectionalLight** node. It is practical to use combination of several such lights to illuminate objects from several sides.

- Adding scene surrounding. This step is not crucial for visualization of interactions between parts in a machine assembly, but it is very important for visualization of aircraft / vehicle dynamics simulation etc., where the position of one object relative to the scene in which it operates is important.

  For instance, if we are to visualize a vehicle dynamics simulation, we need to place a virtual car on a virtual road. Both object need to be to scale (the length units in the car and road models must match), and the car has to be placed in a correct position relative to the road. Proper car scaling, placement and orientation in the scene can be done by defining corresponding fields of the main object **Transform** node mentioned in the "Wrapping Shape Objects by Transforms" section.

An example of a complete scene definition can be found in the **octavia_scene.wrl** VRML file that belongs to the Simulink 3D Animation demo **vr_octavia**.

5.3. Linking the Virtual Scene to a Simulink / SimMechanics / MATLAB Model

The main purpose of this step is to create associations between dynamic model object quantities and corresponding VRML object properties (positions, rotations, etc.) to establish a live data connection between the model and the virtual scene.

Mechanical systems are typically modelled using SimMechanics or Simulink, but Simulink 3D Animation allows also visualization of models implemented in MATLAB. While we will focus on using Simulink 3D Animation Simulink functionality, we will shortly mention also specifics of using the toolbox MATLAB interface.

5.3.1. Linking the Virtual Scene to a Simulink Model

Association of Simulink model signals to virtual scene object properties is accomplished by the **VR Sink** block from the Simulink 3D Animation block library **vrlib**.

In order to associate a Simulink signal to a virtual object property:

- From the **vrlib** library, insert a **VR Sink** block to your Simulink model.
- Double-clik on the **VR Sink** block. This opens the VR Sink Block Parameters dialog, where you can define the associated virtual scene. Enter the name of the scene VRML file in the **Source file:** edit box, or use the **Browse…** button to select the file interactively. Press the **Apply** button to load the selected VRML scene.
- For smooth visualization of the movement it is sometimes necessary to change the block **Sample time**. For instance, in order to update the virtual scene 25 times per simulation second, set the sample time to 0.04 s. Be careful when using the inherited sample time for the **VR Sink** block. Depending on the solver used, this might result in non-equidistant (in simulation time) updating of virtual scene, giving the user false impression of system dynamics.
- In the **VRML Tree** in the right of the dialog box, expand the main object Transform branch and in the scene object hierarchy (displayed exactly as read from the VRML file), locate all parts you want to control from Simulink according to their names given in the "Adding DEF Names" section. With each part (represented by named Transform nodes) select the checkbox next to its **rotation** and **position** fields. This tells VR Sink block that you want to control rotation and position of these parts. You can select also other properties of virtual scene objects (colours etc.), but rotations and positions are most frequently used.
- Press **OK**. For each selected field, VR Sink block creates an input port. Increase the VR Sink block size to reflect the number of input ports.

Once the **VR Sink** block is associated with a virtual scene, double-clicking on it invokes the **Simulink 3D Animation Viewer**. Block parameters are available through **Simulation – Block Properties** menu command in the viewer.

VR Sink inputs take signals of type corresponding to their VRML representation. Position inputs are of type **SFVec3f** – take the position in the [x y z] coordinates. Rotation inputs are of type **SFRotation** – take the 4-element vector defining right-hand rotation as [axis angle]. The angle value is in radians.

It is up to the user to match the coordinate system used by the Simulink model to that of the virtual scene. If the two systems are not identical, some kind of axes transformations is necessary.

While object positions are usually available in the form required by VRML (Cartesian coordinates), rotations have to be usually converted from some other rotation representation. In many cases, object rotations are defined using the rotation matrix representation. For converting such rotations into VRML format, there is the **Rotation Matrix to VRML Rotation** block available in the **vrlib/Utilities** sub-library.

At this point it is again important to distinguish the 2 cases depending on the virtual scene hierarchy:

- All parts in Simulink are defined in global coordinates, virtual scene has a flat structure of independent objects

Object positions: Send to VR Sink all positions in global coordinates.

Object rotations: Send to VR Sink all rotations in global coordinates, with centre of rotation defined in the coordinate system origin.

As the default centre of rotation of VRML Transform objects is [0 0 0], it is usually not necessary to define the centre of rotation for each part in the VRML file specifically.

- All parts in Simulink model follow hierarchical relations, virtual scene has a nested structure

Object positions: Send to VR Sink all positions in local coordinates of (relative to) their parents – predecessors in the object hierarchy. Example – robot's **`tool`** position relative to the robot's **`hand`**.

Object rotations: Send to VR Sink all rotations in local coordinates of (relative to) their parents – predecessors in the object hierarchy. Example – robot's **`tool`** rotation relative to the robot's **`hand`**.

In this case, it is usually necessary to collocate center of rotation defined in the virtual scene with the center of rotation defined in the Simulink model, to match visually

positions of joints between objects. Joints between parts are usually positioned not in the [0 0 0] of parent's coordinate system. To define a center of rotation different from the default value [0 0 0], define the **center** field of the child **Transform** in the VRML file. Example – define the robot's **`tool`** center of rotation to be collocated with the joint connecting the **`hand`** and the **`tool`** , in the hand's local coordinates.

Hint: In the case of hierarchical scene structure, when the parts are connected by revolute joints, it is very easy to define relative rotations between parts. The joint axis defines directly the VRML rotation axis, so constructing the [axis angle] 4-element VRML rotation vector is trivial.

5.3.2. Initial Conditions

Simulink model initial conditions must correspond to the initial object's positions and rotations defined in the virtual scene. Otherwise, the object controlled from Simulink would "jump" from the position defined in the VRML file to the position dictated from Simulink at the start of the simulation. The possible offset can be compensated either in the VRML file (by defining another level of nested **Transform** around the controlled object) or in the Simulink model by adding the object initial position to the model calculations before sending to the **VR Sink** block.

In order to align Simulink model initial conditions with virtual scene object positions, **while maintaining also correct position of the object relative to the surrounding scene**, it is sometimes necessary to adjust also position of the object surroundings. Example – move the road position so that the car at position [0 0 0] stays on the road, the wheels not sinking nor floating above the road surface.

5.3.3. Use of VR Placeholder and VR Signal Expander

The **VR Sink** block accepts only inputs that define fully qualified VRML field values. For dynamic models that describe system's behaviour only in one

dimension, it is still necessary to define full 3D positions for all controlled objects for their virtual reality visualization.

In order to simplify modelling in such cases, Simulink 3D Animation contains the **VR Placeholder** and **VR Expander** blocks.

The **VR Placeholder** block sends out a special value that is interpreted as "unspecified" by the **VR Sink** block. When this placeholder value appears on a VR Sink input, whether as a single value or as an element of a vector, the appropriate value in the virtual world stays unchanged.

The **VR Signal Expander** block creates a vector of predefined length, using some values from the input ports and filling the rest with placeholder signal values.

In order to control position of a virtual object from one-dimensional dynamic model, use the **VR Signal Expander** block with the controlled dimension as its input, its output is 3-component vector that can be used in VR Sink. The remaining vector elements are filled with placeholder signals.

Use of the **VR Signal Expander** block is possible also when defining rotations. When the axis of rotation (as a part of initial rotation of an object **Transform** node) is defined in the VRML file, it is possible to send to the **VR Sink** block a VRML rotation value consisting of 3 placeholder signals and the computed angle, forming a valid 4-element [axis angle] vector.

5.3.4. Linking the Virtual Scene to a SimMechanics Model

SimMechanics is very well suited for 3D visualizations using Simulink 3D Animation. Apart from other features that SimMechanics offers for modelling mechanical assemblies, there are the following features that simplify the visualization of SimMechanics models in virtual reality:

- SimMechanics and VRML coordinate systems are identical (right-handed Cartesian systems with the same orientation of **x**, **y**, **z** azes.)
- In SimMechanics it is easily possible to work with both global and local object coordinates, so it is easy to adapt the model to the structure of the virtual scene exported from the CAD tool.

In addition, SimMechanics offers a convenient way of importing CAD assembly designs into SimMechanics machines – CAD Translator. When the CAD assembly is in parallel exported from the CAD tool also to the VRML model, it is just a matter of few steps to add virtual reality visualization to such machines.

Depending on the virtual scene hierarchy, there are two methods for visualization of SimMechanics machines possible:

- When the virtual scene has a flat structure of independent objects, positions and rotations of machine parts – bodies can be obtained using **Body Sensor** blocks connected to appropriate coordinate systems attached to the bodies, with positions and rotations defined using the absolute – world coordinates. In most cases it is suitable to connect the sensor to a body coordinate system with origin in [0 0 0] and initial rotation matrix defined as identity matrix [1 0 0; 0 1 0, 0 0 1] in the world coordinates.
- When the virtual scene has a hierarchical structure of nested objects, the body positions and rotations can be obtained using **Body Sensor** blocks with output set to local body coordinates. In some special cases – for instance when two bodies are connected through a revolute joint, it is possible to get the angle between the objects using a **Joint Sensor** block.

5.3.5. Linking the Virtual Scene to a MATLAB Model

For interaction with virtual scenes Simulink 3D Animation offers also set of MATLAB functions and constructs known as the toolbox MATLAB Interface. There might be circumstances when using this MATLAB functionality is suitable in connection with CAD-based designs, for instance:

- Using customised GUIs to visualize static objects or their relations in virtual environment. Example – interactive machine assembly instructions.
- Visualization of a 3D information based on a general independent quantity (doesn’t have to be time).
- MATLAB interface functions used in Simulink model callbacks.
- Visualization of systems whose dynamic models are available as MATLAB code.

- Visualization of systems where massive object changes eg. deformations take place. In this case, sending dynamically-sized matrix-type data from the dynamic models to virtual scenes is necessary.

Seznam použité literatury

[1] International Standard ISO/IEC 14772-1:1997, the Virtual Reality Modeling Language (VRML); http://new.web3d.org/files/specifications/14772/V2.0/index.html

[2] Žára, Jiří: Laskavý průvodce virtuálními světy; http://dcgi.felk.cvut.cz/cgg/LaskavyPruvodce

[3] Marrin, Chris, Campbell, Bruce: Teach Yourself VRML 2 in 21 days; Sams.net,1997

[4] Dokumentace k produktu Simulink 3D Animation; http://www.mathworks.com/help/toolbox/sl3d

PŘÍLOHY

Centrum pro rozvoj výzkumu pokročilých řídicích a senzorických technologií
CZ.1.07/2.3.00/09.0031

Ústav automatizace a měřicí techniky
VUT v Brně
Kolejní 2906/4
612 00 Brno
Česká Republika

http://www.crr.vutbr.cz

info@crr.vutbr.cz